National Institute of
Standards and Technology
U.S. Department of Commerce

Special Publication 800-126
Revision 2 (Draft)

The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2 (Draft)

Recommendations of the National Institute of Standards and Technology

David Waltermire
Stephen Quinn
Karen Scarfone
Adam Halbardier

NIST Special Publication 800-126
Revision 2 (Draft)

The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2 (Draft)

Recommendations of the National Institute of Standards and Technology

David Waltermire
Stephen Quinn
Karen Scarfone
Adam Halbardier

C O M P U T E R S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

July 2011

U.S. Department of Commerce

Gary Locke, Secretary

National Institute of Standards and Technology

Dr. Patrick D. Gallagher, Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Special Publication 800-series reports on ITL's research, guidance, and outreach efforts in computer security and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Special Publication 800-126, Revision 2 (Draft)
51 pages (Jul. 2011)

Acknowledgments

The authors, David Waltermire and Stephen Quinn of the National Institute of Standards and Technology (NIST) Karen Scarfone of Scarfone Cybersecurity, and Adam Halbardier of Booz Allen Hamilton wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content.

The authors would like to acknowledge the following contributors for their keen and insightful assistance with developing the current and previous versions of this specification: John Banghart, Harold Booth, Paul Cichonski, and Blair Heiserman of NIST; Christopher Johnson of HP Enterprise Services; Paul Bartock of the National Security Agency (NSA); Jeff Ito, Matt Kerr, Shane Shaffer, and Greg Witte of G2, Inc.; Andy Bove of SecureAcuity; Jim Ronayne of Varen Technologies; Rhonda Farrell, Angela Orebaugh, and Victoria Thompson of Booz Allen Hamilton; Alan Peltzman of the Defense Information Systems Agency (DISA); and Jon Baker, Drew Buttner, Maria Casipe, and Charles Schmidt of the MITRE Corporation.

Trademark Information

Table of Contents

List of Tables and Figures

Executive Summary

The Security Content Automation Protocol (SCAP) is a suite of specifications that standardize the format and nomenclature by which security software products communicate security content, particularly software flaw and security configuration information[1]. SCAP is a multi-purpose protocol that supports automated configuration, vulnerability, and patch checking, technical control compliance activities, and security measurement. Goals for the development of SCAP include standardizing system security management, promoting interoperability of security products, and fostering the use of standard expressions of security content.

SCAP Version 1.2 is comprised of eleven component specifications grouped into five categories:

- **Languages.** The SCAP languages provide standard vocabularies and conventions for expressing security policy, technical check mechanisms, and assessment results. The SCAP language specifications are Extensible Configuration Checklist Description Format (XCCDF), Open Vulnerability and Assessment Language (OVAL®), and Open Checklist Interactive Language (OCIL™).

- **Reporting Formats.** The SCAP reporting formats provide the necessary constructs to express collected information in standardized formats. The SCAP reporting format specifications are Asset Reporting Format (ARF) and Asset Identification. Although Asset Identification is not explicitly a reporting format, SCAP uses it as a key component in identifying the assets that reports relate to.

- **Enumerations.** Each SCAP enumeration defines a standard nomenclature (naming format) and an official dictionary or list of items expressed using that nomenclature. The SCAP enumeration specifications are Common Platform Enumeration (CPE™), Common Configuration Enumeration (CCE™), and Common Vulnerabilities and Exposures (CVE®).

- **Measurement and scoring systems.** In SCAP this refers to evaluating specific characteristics of a security weakness (for example, software vulnerabilities and security configuration issues) and, based on those characteristics, generating a score that reflects the relative severity. The SCAP measurement and scoring system specifications are Common Vulnerability Scoring System (CVSS) and Common Configuration Scoring System (CCSS).

- **Integrity.** An SCAP integrity specification helps to preserve the integrity of SCAP content and results. The Trust Model for Security Automation Data (TMSAD) is the SCAP integrity specification.

SCAP utilizes software flaw and security configuration standard reference data. This reference data is provided by the National Vulnerability Database (NVD),[2] which is managed by NIST and sponsored by the Department of Homeland Security (DHS).

This publication defines the technical composition of SCAP Version 1.2 in terms of its component specifications, their interrelationships and interoperation, and the requirements for SCAP content. The technical specification for SCAP in this publication describes the requirements and conventions that are to be employed to ensure the consistent and accurate exchange of SCAP-conformant content and the ability to reliably use the content with SCAP-conformant products.

---

1 Products implementing SCAP can also be used to support non-security use cases such as configuration management and software inventory.

2 The National Vulnerability Database can be found at http://nvd.nist.gov/.

The U.S. Federal Government, in cooperation with academia and private industry, is adopting SCAP and encourages its use in support of security automation activities and initiatives.[3] SCAP has achieved widespread adoption by major software manufacturers and has become a significant component of large information security management and governance programs. The protocol is expected to evolve and expand in support of the growing needs to define and measure effective security controls, assess and monitor ongoing aspects of that information security, and successfully manage systems in accordance with risk management frameworks such as NIST Special Publication 800-53[4], Department of Defense (DoD) Instruction 8500.2, and the Payment Card Industry (PCI) framework.

By detailing the specific and appropriate usage of the SCAP 1.2 components and their interoperability, NIST encourages the creation of reliable and pervasive SCAP content and the development of a wide array of products that leverage SCAP.

Organizations that develop SCAP 1.2-based content or products should comply with the following recommendations:

Follow the requirements listed in this document and in the associated component specifications.

Organizations should ensure that their implementation and use of SCAP 1.2 is compliant with the requirements detailed in each component specification and the information presented in this document.

If requirements are in conflict between component specifications, this document will provide clarification. If a component specification is in conflict with this document, the requirements in this document take precedence.

When creating SCAP content, adhere to the conventions specified in this document.

Security products and checklist authors assemble content from SCAP data repositories to create viable SCAP-conformant security guidance. For example, a security configuration checklist can document desired security configuration settings, installed patches, and other system security elements using a standardized SCAP format. Such a checklist would use XCCDF to describe the checklist, CCE to identify security configuration settings to be addressed or assessed, and CPE to identify platforms for which the checklist is valid. The use of CCE and CPE entries within XCCDF checklists is an example of an SCAP convention—a requirement for valid SCAP usage. These conventions are considered part of the definition of SCAP 1.2. Organizations producing SCAP content should adhere to these conventions to ensure the highest degree of interoperability. NIST provides an SCAP Content Validation Tool that organizations can use to help validate the correctness of their SCAP content. The tool checks that SCAP source and result content is well-formed, all cross references are valid, and required values are appropriately set.[5]

[3] Refer to http://www.whitehouse.gov/omb/memoranda/fy2008/m08-22.pdf.

[4] The Risk Management Framework is described in Section 3.0 of NIST Special Publication 800-53, available at http://csrc.nist.gov/publications/PubsSPs.html#800-53.

[5] http://scap.nist.gov/revision/1.2/#tools

1. Introduction

1.1 Authority

The National Institute of Standards and Technology (NIST) developed this document in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets; but such standards and guidelines shall not apply to national security systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), "Securing Agency Information Systems," as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This guideline has been prepared for use by Federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright, though attribution is desired.

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority, nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official.

1.2 Purpose and Scope

This document provides the definitive technical specification for version 1.2 of the Security Content Automation Protocol (SCAP). SCAP (pronounced ess-cap) consists of a suite of specifications for standardizing the format and nomenclature by which security software communicates information about software flaws and security configurations. This document defines requirements for creating and processing SCAP content. These requirements build on the requirements defined within the individual SCAP component specifications. Each new requirement pertains either to using multiple component specifications together or to further constraining one of the individual component specifications. The requirements within the individual component specifications are not repeated in this document; see those specifications to access their requirements.

The scope of this document is limited to SCAP version 1.2. Other versions of SCAP and its component specifications, including emerging specifications, are not addressed here. Future versions of SCAP will be defined in distinct revisions of this document, each clearly labeled with a document revision number and the appropriate SCAP version number. SCAP revisions are managed through a coordinated process defined within the SCAP Release Cycle.[6] The release cycle workflow manages changes related to SCAP specifications and validation processes including the addition of new specifications or updates to existing specifications. This process encourages community involvement, promotes transparency and awareness regarding proposed changes, and affords ample lead time to prepare for pending changes.

[6] SCAP Release Cycle, http://scap.nist.gov/timeline.html

1.3 Audience

This document is intended for three primary audiences:

- Content authors and editors seeking guidance to ensure that the SCAP content they produce operates correctly, consistently, and reliably in SCAP products.
- Software developers and system integrators seeking to create, use, or exchange SCAP content in their products or service offerings.
- Product developers preparing for SCAP validation at an accredited independent testing laboratory.

This document assumes that readers already have general knowledge of SCAP and reasonable familiarity with the SCAP component specifications that their content, products, or services use. Individuals without this level of knowledge who would like to learn more about SCAP should consult NIST Special Publication (SP) 800-117, *Guide to Adopting and Using the Security Content Automation Protocol.*[7]

1.4 Document Structure

The remainder of this document is organized into the following major sections and appendices:

- Section 2 provides the high-level requirements for claiming conformance with the SCAP 1.2 specification.
- Section 3 details the requirements and recommendations for SCAP content syntax, structures, and development.
- Section 4 defines SCAP content processing requirements and recommendations.
- Section 5 provides additional requirements for particular use cases.
- Appendix A contains an acronym and abbreviation list.
- Appendix B lists references and other resources related to SCAP 1.2.
- Appendix C provides a change log that documents significant changes to major drafts of this specification.

1.5 Document Conventions

Some of the requirements and conventions used in this document reference Extensible Markup Language (XML) content [XMLS]. These references come in two forms, inline and indented. An example of an inline reference is: a `<cpe_dict:cpe-item>` may contain `<cpe_dict:check>` elements that reference OVAL Definitions.

In this example the notation `<cpe_dict:cpe-item>` can be replaced by the more verbose equivalent "the XML element whose qualified name is `cpe_dict:cpe-item`".

[7] http://csrc.nist.gov/publications/PubsSPs.html#800-117

An example of an indented reference is:

> References to OVAL Definitions are expressed using the following format:
>
> `<cpe_dict:check system=`
> `"http://oval.mitre.org/XMLSchema/oval-definitions-5"`
> `href="`*`Oval_URL`*`">[`*`Oval_inventory_definition_id`*`]`
> `</cpe_dict:check>`.

The general convention used when describing XML attributes within this document is to reference the attribute as well as its associated element including the namespace alias, employing the general form "*`@attributeName`* for the *`<prefix:localName>`*".

Indented references are intended to represent the form of actual XML content. Indented references represent literal content by the use of a `fixed-length font`, and parametric (freely replaceable) content by the use of an *italic font*. Square brackets '`[]`' are used to designate optional content. Thus "`[`*`Oval_inventory_definition_id`*`]`" designates optional parametric content.

Both inline and indented forms use qualified names to refer to specific XML elements. A qualified name associates a named element with a namespace. The namespace identifies the XML model, and the XML schema is a definition and implementation of that model. A qualified name declares this schema to element association using the format '*prefix*:*element-name*'. The association of prefix to namespace is defined in the metadata of an XML document and varies from document to document. In this specification, the conventional mappings listed in Table 1 are used. The namespaces in the table are not required to be resolvable URIs; if you enter them into a web browser, for example, they may or may not work.[8]

Table 1. Conventional XML Mappings

Prefix	Namespace	Schema
ai	http://scap.nist.gov/schema/asset-identification/1.1	Asset Identification
arf	http://scap.nist.gov/schema/asset-reporting-format/1.1	ARF
arf-rel	http://scap.nist.gov/vocabulary/arf/relationships/1.0#	ARF relationships
cat	urn:oasis:names:tc:entity:xmlns:xml:catalog	XML Catalog
cpe	http://cpe.mitre.org/language/2.0	Embedded CPE references
cpe-dict	http://cpe.mitre.org/dictionary/2.0	CPE dictionaries
cve	http://scap.nist.gov/schema/vulnerability/0.4	NVD/CVE data feed elements and attributes
cvss	http://scap.nist.gov/schema/cvss-v2/0.2	NVD/CVSS data feed elements and attributes
dc	http://purl.org/dc/elements/1.1/	Simple Dublin Core elements
ds	http://scap.nist.gov/schema/scap/source/1.2	SCAP source data stream collection
dt	http://scap.nist.gov/schema/xml-dsig/1.0	Security automation digital signature extensions
nvd	http://scap.nist.gov/schema/feed/vulnerability/2.0	Base schema for NVD data feeds
ocil	http://scap.nist.gov/schema/ocil/2.0	OCIL elements and attributes
oval	http://oval.mitre.org/XMLSchema/oval-common-5	Common OVAL elements and attributes
oval-def	http://oval.mitre.org/XMLSchema/oval-definitions-5	OVAL definitions

[8] Regarding a namespace URI, the W3C document titled "Namespaces in XML 1.0 (Third Edition) states that "it is not a goal that it be directly usable for retrieval of a schema (if any exists)." http://www.w3.org/TR/xml-names/#A785

Prefix	Namespace	Schema
oval-res	http://oval.mitre.org/XMLSchema/oval-results-5	OVAL results
oval-sc	http://oval.mitre.org/XMLSchema/oval-system-characteristics-5	OVAL system characteristics
oval-var	http://oval.mitre.org/XMLSchema/oval-variables-5	The elements, types, and attributes that compose the core schema for encoding OVAL Variables. This schema is provided to give structure to any external variables and their values that an OVAL Definition is expecting.
scap-rel	http://scap.nist.gov/vocabulary/scap/relationships/1.0#	SCAP relationships
sch	http://purl.oclc.org/dsdl/schematron	Schematron validation scripts
xccdf	http://checklists.nist.gov/xccdf/1.2	XCCDF policy documents
xml	http://www.w3.org/XML/1998/namespace	Common XML attributes
xxxx-def	http://oval.mitre.org/XMLSchema/oval-definitions-5#xxxx	OVAL elements and attributes specific to an OS, Hardware, or Application type *xxxx*[9]
xxxx-sc	http://oval.mitre.org/XMLSchema/oval-system-characteristics-5#xxxx	OVAL system characteristic elements and attributes specific to an OS, Hardware, or Application type *xxxx*

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in Request for Comment (RFC) 2119.[10]

[9] The types supported by OVAL 5.3 include the AIX, CATOS, ESX, FREE BSD, HP-UX, IOS, LINUX, PIXOS, SOLARIS, UNIX, WINDOWS, INDEPENDENT (common) operating systems, and APACHE application.

[10] RFC 2119, “Key words for use in RFCs to Indicate Requirement Levels”, is available at http://www.ietf.org/rfc/rfc2119.txt.

2. SCAP 1.2 Conformance

SCAP 1.2 uses the following specifications:

- Languages
  - Extensible Configuration Checklist Description Format (XCCDF) 1.2, a language for authoring security checklists/benchmarks and for reporting results of evaluating them [XCCDF]
  - Open Vulnerability and Assessment Language (OVAL) 5.10, a language for representing system configuration information, assessing machine state, and reporting assessment results [OVAL]
  - Open Checklist Interactive Language (OCIL) 2.0, a language for representing checks that collect information from people or from existing data stores made by other data collection efforts [OCIL]
- Reporting Formats
  - Asset Reporting Format (ARF) 1.1, a format for expressing the transport format of information about assets and the relationships between assets and reports [ARF]
  - Asset Identification 1.1, a format for uniquely identifying assets based on known identifiers and/or known information about the assets [AI]
- Enumerations
  - Common Platform Enumeration (CPE) 2.3, a nomenclature and dictionary of hardware, operating systems, and applications [CPE]
  - Common Configuration Enumeration (CCE) 5, a nomenclature and dictionary of software security configurations [CCE]
  - Common Vulnerabilities and Exposures (CVE), a nomenclature and dictionary of security-related software flaws[11] [CVE]
- Measurement and Scoring Systems
  - Common Vulnerability Scoring System (CVSS) 2.0, a specification for measuring the relative severity of software flaw vulnerabilities [CVSS]
  - Common Configuration Scoring System (CCSS) 1.0, a specification for measuring the relative severity of system security configuration issues [CCSS]
- Integrity
  - Trust Model for Security Automation Data (TMSAD) 1.0, a specification for using digital signatures in a common trust model applied to security automation specifications [TMSAD].

All references to these specifications within this document are to the version numbers listed above, unless otherwise explicitly specified.

Combinations of these specifications can be used together for particular functions, such as security configuration checking. These functions, known as *SCAP use cases*, are ways in which a product can use SCAP. The collective XML content used for a use case is called an *SCAP data stream*, which is a specific

[11] CVE does not have a version number.

instantiation of SCAP content. An *SCAP source data stream* holds the input content, and an *SCAP result data stream* holds the output content. The major elements of a data stream, such as the XCCDF portion or the OVAL patch portion, are referred to as *stream components*.

Products and organizations may want to claim conformance to one or more of the SCAP use cases within the SCAP 1.2 specification for a variety of reasons. For example, a product may want to assert that it uses SCAP content properly and can interoperate with other products using valid SCAP content. Another example is a policy mandating that an organization use SCAP for performing vulnerability assessments and other security operations.

This section provides the high-level requirements that a product or content must meet for conformance with the SCAP 1.2 specification. Such products and content are referred to as *SCAP conformant.* Most of the requirements listed in this section reference other sections in the document that fully define the requirements.

2.1 Product Conformance

There are two types of SCAP-conformant products: content producers and content consumers. *Content producers* are products that generate SCAP source data stream content, while *content consumers* are products that accept existing SCAP source data stream content, process it, and produce SCAP result data streams. Products claiming conformance with the SCAP 1.2 specification SHALL comply with the following requirements:

1. Adhere to the requirements detailed in each applicable component specification (for each selected SCAP component specification, and for each SCAP component specification required to implement the selected SCAP use cases). The authoritative references for each specification are listed in Appendix B. **If requirements are in conflict between component specifications, this document will provide clarification. If a component specification is in conflict with this document, the requirements in this document SHALL take precedence.**

2. Adhere to the requirements detailed in the errata for this document [ERRATA]. **If requirements are in conflict between the errata and this document, the errata SHALL take precedence.**

3. For content producers, generate well-formed SCAP source data streams. This includes following the content conformance requirements specified in Section 2.2. This also includes following the requirements in Section 5 for the use cases that the content producer supports.

4. For content consumers, consume and process well-formed SCAP source data streams, and generate well-formed SCAP result data streams. This includes following all of the processing requirements defined in Section 4 for each selected SCAP component specification and each SCAP component specification required to implement the selected SCAP use cases.

5. Make an explicit claim of conformance to this specification in any documentation provided to end users.

2.2 Source Content Conformance

Source content (i.e., source data streams) claiming conformance with the SCAP 1.2 specification SHALL comply with the following requirements:

1. Adhere to the requirements detailed in each applicable component specification (for each selected SCAP component specification, and each SCAP component specification required to implement the selected SCAP use cases). The authoritative references for each specification are listed in Appendix B. **If requirements are in conflict between component specifications, this document will provide clarification. If a component specification is in conflict with this document, the requirements in this document SHALL take precedence.**

2. Adhere to the requirements detailed in the errata for this document [ERRATA]. **If requirements are in conflict between the errata and this document, the errata SHALL take precedence.**

3. Follow all of the syntax, structural, and other source content design requirements defined in Section 3 for each selected SCAP component specification and for each SCAP component specification required to implement the selected SCAP use cases. Also, follow all of the requirements specified for the content's use cases as defined in Section 5.

3. SCAP Content Requirements and Recommendations

This section defines the SCAP 1.2 content syntax, structure, and development requirements that SCAP-conformant products and content MUST follow. This section also provides recommendations that are not mandatory; organizations are encouraged to adopt them to promote stronger interoperability and greater content consistency. The first part of the section discusses SCAP source data stream requirements and source components. The middle of the section groups requirements and recommendations by specification: XCCDF, OVAL, OCIL, CPE, CCE, CVE, CVSS, and CCSS, in that order. Finally, the last part of the section discusses use of XML digital signatures for source data stream content.

3.1 SCAP Source Data Stream

An *SCAP data stream* is a bundle of SCAP components along with the mappings of references between SCAP components. (Section 3.2 defines the various SCAP components.) There are two types of SCAP data streams: source and result. SCAP source data streams are discussed in this section, while SCAP result data streams are discussed in Section 4.4 as part of the requirements for SCAP processing.

An *SCAP source data stream collection* is composed of two sections: the SCAP data streams and the SCAP components. The components section contains an unbounded number of components. The data streams section contains one or more data streams, each of which references the components in the components section that compose it. This model allows components to be reused across data streams. Multiple data streams are allowed in a data stream collection to allow grouping of related or similar source data streams. For example, NIST currently distributes the United States Government Configuration Baseline (USGCB)[12] as a series of SCAP bundles. In the future, source data streams that are similar or related (e.g., Microsoft Windows 7 content and Microsoft Windows 7 Firewall content) could be bundled into the same source data stream collection. Figure 1 shows the relationship between data stream collections, data streams, and components.

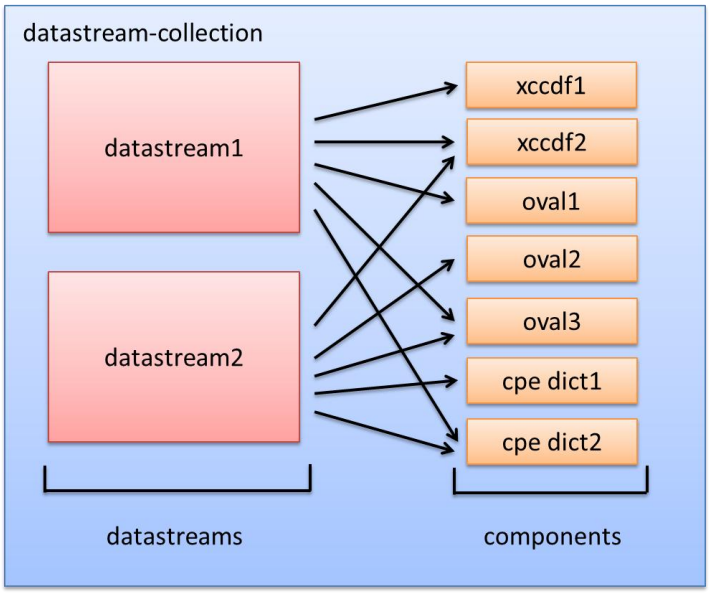

Figure 1 - SCAP Data Stream Collection

[12] http://usgcb.nist.gov/

In Figure 1, datastream1 points to xccdf1, xccdf2, oval1, oval3, and cpe dict2. datastream2 points to xccdf2, oval2, oval3, cpe dict1, and cpe dict2. The data streams are composed of links to the components that they reference; each logical link encapsulates the information required to allow the content consumer to connect the components together within the data stream. Content authors MAY place components in any order. For example, some authors might choose to place dictionary components first to help optimize data stream parsing.

Each data stream is a collection of links to other components. Links serve two purposes: to indicate which component is being referred to, and to provide a map to dereference associations external to the component. The latter enables a data stream to define context for the referenced component relative to the link that is pointing to it. Figure 2 shows a sample data stream.

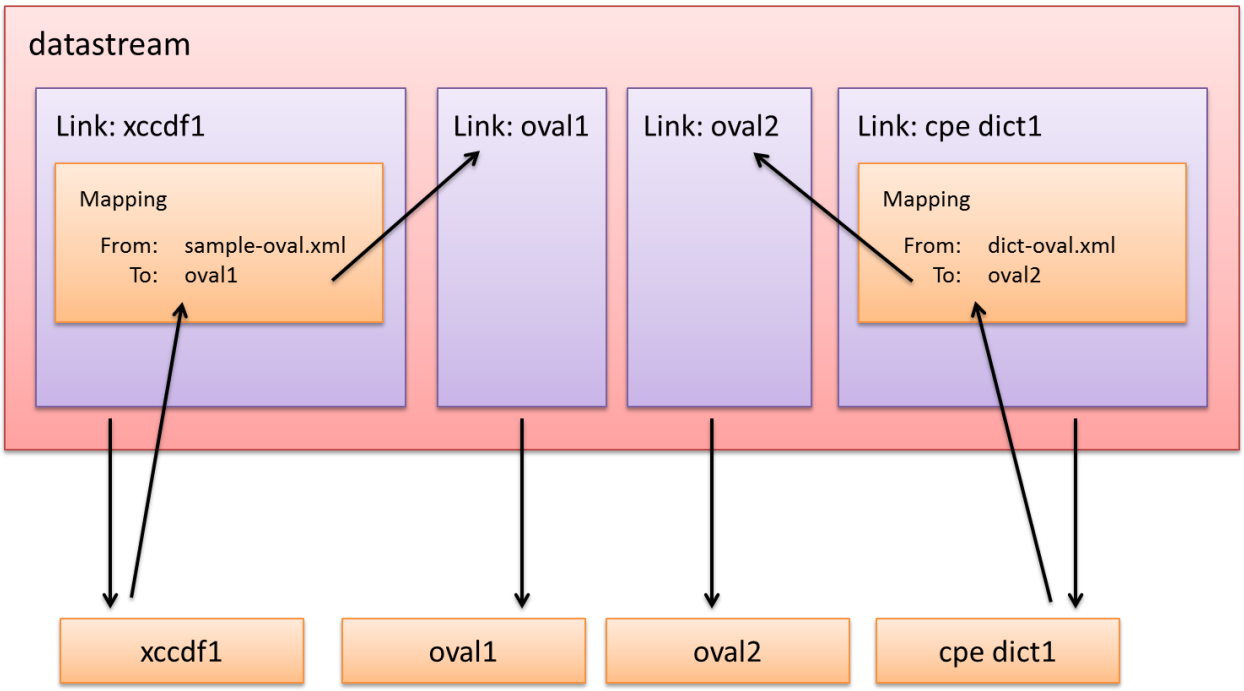

Figure 2 - SCAP Data Stream

In Figure 2, the data stream links to four components. The OVAL components do not reference out to external content, so there are no mappings captured for them. The XCCDF and CPE Dictionary content do link out to external content, so whenever referencing those components, a mapping is created. In the figure, the xccdf1 content creates a mapping that indicates that whenever xccdf1 references "sample-oval.xml", the content is found through the oval1 link. Similarly, when the cpe dict1 component references "dict-oval.xml", the content is found through the oval2 link. This approach allows components to be reused across data streams and data stream collections, and to be bound together at the SCAP logical level instead of at the component level.

The design of the SCAP source data stream is important for the following reasons:

1. Individual components may be developed outside of an SCAP data stream where the binding to other components is not necessarily known at the time the component is created.

2. The SCAP source data stream creates a binding between different components that were not necessarily designed to reference each other.

3. The logical link mapping in the data stream places a layer of capability within the data stream to control the dereferencing of URIs within components, creating a complete solution related to bundling components.

4. The SCAP source data stream format will be useful in future communication models such as web services, transport protocols, tasking mechanisms, etc.

5. The SCAP source data stream format supports more comprehensive validation of component content, including interrelationships between components.

The following tables formalize the SCAP source data stream data model. The data contained in the tables are requirements and MUST be interpreted as follows:

- The "Element Name" field indicates the name for the XML element being described. Each element name has a namespace prefix indicating the namespace to which the element belongs. See Table 1 for a mapping of namespace prefixes to namespaces.
- The "Definition" field indicates the prose description of the element. The definition field MAY contain requirement words as indicated in [RFC 2119].
- The "Properties" field is broken into four subfields:
  - The "Name" column indicates the name of a property that MAY or MUST be included in the described element, in accordance with the cardinality indicated in the "Count" field.
  - The "Type" column indicates the REQUIRED data type for the value of the property. There are two categories of types: literal and element. A literal type indicates the type of literal as defined in [XMLS]. An element type references the name of another element that ultimately defines that property.
  - The "Count" column indicates the cardinality of the property within the element. The property MUST be included in the element in accordance with the cardinality. If a range is given, and "n" is the upper bound of the range, then the upper limit SHALL be unbounded.
  - The "Definition" column defines the property in the context of the element. The definition MAY contain requirement words as indicated in [RFC 2119].

Table 2 - ds:data-stream-collection

Element Name: ds:data-stream-collection				
Definition	The root element for an SCAP data stream collection. This element is the top-level element of a data stream collection. It contains the data streams and components that comprise this data stream collection, along with any data stream signatures.			
Properties	**Name**	**Type**	**Count**	**Definition**
	id	literal – ID	1	The identifier for the data stream collection. This identifier MUST be globally unique.
	data-stream	element – ds:data-stream	1-n	An element that represents a single data stream.
	component	element – ds:component	1-n	An element that represents content expressed using an SCAP component specification.
	extended-component	element – ds:extended-component	0-n	An element that holds additional components to enable extension.
	Signature	element – dsig:Signature	0-n	An XML digital signature element. Sections 3.11 and 4.8 define the requirements for this element.

Table 3 - ds:data-stream

Element Name: ds:data-stream				
Definition	A data stream. This element contains the links to all of the components that comprise this data stream.			
Properties	**Name**	**Type**	**Count**	**Definition**
	id	literal – ID	1	The identifier for the data stream. This identifier MUST be globally unique.
	use-case	literal – token	1	The use case represented by the data stream. The value MUST be one of the following: CONFIGURATION, VULNERABILITY, OVAL_ONLY, or INVENTORY. The value selected MUST indicate which type of content is being represented as defined in Section 5.
	scap-version	literal – token	1	The targeted SCAP version. The value MUST be 1.2, 1.1, or 1.0. The value MUST indicate which version of SCAP the content is conformant with. 1.2 MUST be specified to be conformant with this version of SCAP.
	timestamp	literal – dateTime	0-1	The date and time when this datastream was created.
	dictionaries	element – ds:dictionaries	0-1	Links to dictionary content.
	checklists	element – ds:checklists	0-1	Links to checklist content.
	checks	element – ds:checks	1	Links to check content.
	extended-components	element – ds:extended-components	0-1	Links to non-standard components captured as `<ds:extended-component>` elements. Products that do not understand the contained reference will ignore it. The `<ds:extended-component>` MAY contain an `<xccdf:tailoring>` element as its immediate descendant; in all other cases, linking to a `<ds:extended-component>` SHALL make the data stream nonconformant with SCAP.

Table 4 - ds:dictionaries

Element Name: ds:dictionaries				
Definition	A container element that holds references to one or more dictionary components.			
Properties	**Name**	**Type**	**Count**	**Definition**
	component-ref	element – component-ref	1-n	MUST contain a reference to a dictionary component (a component containing CPE dictionary content).

Table 5 - ds:checklists

Element Name: ds:checklists				
Definition	A container element that holds references to one or more checklists.			
Properties	**Name**	**Type**	**Count**	**Definition**
	component-ref	element – component-ref	1-n	MUST contain a reference to a checklist component (a component containing an <xccdf:Benchmark> element).

Table 6 - ds:checks

Element Name: ds:checks				
Definition	A container element that holds references to one or more check components.			
Properties	**Name**	**Type**	**Count**	**Definition**
	component-ref	element – component-ref	1-n	MUST contain a reference to a check component (a component containing OVAL or OCIL content).

Table 7 - ds:extended-components

Element Name: ds:extended-components				
Definition	A container element that holds references to one or more extended components.			
Properties	**Name**	**Type**	**Count**	**Definition**
	component-ref	element – component-ref	1-n	Intended to capture additional components in the SCAP data stream. This field MUST contain a reference to a <ds:extended-component>. See Table 3 for additional information.

Table 8 - ds:component-ref

Element Name: ds:component-ref				
Definition	An element that encapsulates the information necessary to link to a component within the data stream collection, or to external content, which gives context to the reference. This is a simple XLink [XLINK].			
Properties	**Name**	**Type**	**Count**	**Definition**
	id	literal - ID	1	The identifier for the reference. This identifier MUST be globally unique.
	type	literal – xlink:type	0-1	The type of XLink represented. The <ds:component-ref> is constrained to a simple XLink, so the value of this field MUST be 'simple' if specified.
	href	literal – xlink:href	1	A URI to the target component (either local to the data stream collection or remote). When referencing a local component, the URI MUST be in the form '#' + componentId (e.g. "#component1"). When referencing external content, the URI MUST dereference to an XML stream representing the content of the target component.
	catalog	element – cat:catalog	0-1	An XML catalog that defines the mapping between external URI links in the component being referenced by this <ds:component-ref>, and where those URIs should map to within the context of this data stream.

Table 9 – cat:catalog

Element Name: cat:catalog				
Definition	A catalog element defined by the OASIS XML Catalog specification [XMLCAT]. Within an SCAP source data stream this element SHALL contain one or more `<cat:uri>` and/or `<cat:rewriteURI>` elements, and it SHALL NOT contain any other elements or attributes.			
Properties	**Name**	**Type**	**Count**	**Definition**
	uri	element – cat:uri	0-n (at least 1 of this or rewriteURI MUST be provided)	A `<cat:uri>` element. It SHALL have a `@name` attribute and a `@uri` attribute specified. The `@name` attribute SHALL be populated with the URI of an external link specified within the component referenced by this element's parent `<ds:component-ref>` element. The `@uri` attribute SHALL be populated with a URI of the `<ds:component-ref>` that links to the component that MUST be resolved when the name URI is found in the referenced component.
	rewriteURI	element – cat:rewriteURI	0-n (at least 1 of this or uri MUST be provided)	A `<cat:rewriteURI>` element. It SHALL have a `@uriStartString` attribute and a `@rewritePrefix` attribute specified. The `@uriStartString` attribute SHALL be populated with the start of a URI of an external link specified within the component referenced by this element's parent `<ds:component-ref>` element that is to be replaced. The `@rewritePrefix` attribute SHALL be populated with a string that will replace the matched `@uriStartString` value. The resulting URI MUST be used to resolve the link. See [XMLCAT] for more details.

Table 10 – ds:component

Element Name: ds:component				
Definition	This element holds a single component. The types of components are defined in Section 3.2.			
Properties	**Name**	**Type**	**Count**	**Definition**
	id	literal – ID	1	The identifier for the component. This identifier MUST be globally unique.
	timestamp	literal – dateTime	1	Indicates when the component was created or last updated.
	Benchmark	xccdf:Benchmark	1, and only 1, of these elements	An XCCDF benchmark
	oval_definitions	oval-def:oval_definitions		OVAL definitions
	ocil	ocil:ocil		OCIL content
	cpe-list	cpe-dict:cpe-list		A CPE list

Table 11 – ds:extended-component

Element Name: ds:extended-component				
Definition	This element holds content that does not fit within the defined component types. Authors SHOULD use this element as an extension point to capture content that is not captured in a regular component. The content of this element SHALL be an XML element in a namespace other than the SCAP source data stream namespace.			
Properties	**Name**	**Type**	**Count**	**Definition**
	id	literal – ID	1	The identifier for the component. This identifier MUST be globally unique.
	timestamp	literal – dateTime	1	Indicates when the component was created or last updated.

The SCAP source data stream collection SHALL validate against the XML schema representation for the source data stream, which is available at http://scap.nist.gov/revision/1.2/#schema.

Each `<ds:component>` and `<ds:extended-component>` element SHALL validate against the corresponding component schema.

If applicable, the SCAP source data stream collection and each component SHOULD validate against the associated Schematron stylesheet. NIST provides and maintains a set of Schematron rules to check well-formed SCAP content. Content SHOULD pass all Schematron assertions in the Schematron rule files. Failed assertions with a "warning" flag MAY be disregarded if the assertion discovers an issue in the content that is justifiable and expected based on the needs of the content author. All other failed assertions SHOULD be resolved. The Schematron files are located at http://scap.nist.gov/revision/1.2/#schematron. The SCAP Schematron rule set is an interpretation of this specification. The implementation of the rules is subject to change.

3.2 SCAP Source Components

An SCAP source data stream is the expression of content for a specific use case using one or more stream components. Each SCAP source data stream component SHALL use the element specified in Table 12 as its document element.

Table 12 - SCAP Source Data Stream Conventions

Component	Document Element
XCCDF Benchmark	`<xccdf:Benchmark>`
OVAL	`<oval-def:oval_definitions>`
OCIL Questionnaire	`<ocil:ocil>`
CPE Dictionary	`<cpe-dict:cpe-list>`

Each SCAP source data stream component SHOULD NOT use any constructs that are deprecated in its associated specification. Validation of each component SHALL be done in accordance with the portions of this document that define requirements for the component. NIST provides an SCAP Content Validation Tool, which is designed to help validate the correctness of SCAP data streams.[13] The SCAP Content Validation Tool is a command-line tool that will check that SCAP source and result content is well-

[13] The tool can be downloaded from http://scap.nist.gov/revision/1.2/#tools.

formed, cross references are valid, and required values are appropriately set. Errors and warnings are returned in both XML and Hypertext Markup Language (HTML) formats.

3.3 Extensible Configuration Checklist Description Format (XCCDF)

This section lists the Extensible Configuration Checklist Description Format (XCCDF) requirements and recommendations. They are organized by the following categories: general, CPE names, *<xccdf:Benchmark>*, *<xccdf:Profile>*, check system usage, *<xccdf:Rule>*, and *<xccdf:Value>*.

3.3.1 General

The use of the *@xml:base* attribute SHALL NOT be allowed in SCAP XCCDF content. This attribute is not compatible with the SCAP data stream model.

XCCDF metadata is used by SCAP products to assist in the selection of the appropriate SCAP data stream, ensure that the most recent or correct version of an XCCDF document is used, and provide additional information about the document. The following metadata requirements and conventions apply to the *<xccdf:Benchmark>*, *<xccdf:Profile>*, *<xccdf:Value>*, *<xccdf:Group>*, and *<xccdf:Rule>* elements:

1. One or more instances of the *<xccdf:title>* element SHALL be provided. Each instance MUST contain a text value that indicates the purpose of the containing element.

2. One or more instances of the *<xccdf:description>* element SHALL be provided. Each instance MUST contain a text value that describes the purpose of the containing element.

All remaining OPTIONAL elements in the XCCDF schema[14] MAY be included at the author's discretion unless otherwise noted in this document.

3.3.2 CPE Names

Each CPE name [CPE-N] in an *<xccdf:platform>* or *<cpe-lang:fact-ref>* element within an XCCDF document SHALL match at least one CPE entry in a dictionary referenced by the data stream. A match is considered an EQUAL or SUPERSET result when matching the CPE name to a dictionary entry, as defined in the CPE Matching specification [CPE-M]. Only non-deprecated names SHOULD be used.

3.3.3 The <xccdf:Benchmark> Element

The following requirements and recommendations apply to the *<xccdf:Benchmark>* element:

1. The REQUIRED *@id* attribute SHALL be used to uniquely identify all revisions of a benchmark. Multiple revisions of a single benchmark SHOULD have identical identifiers, so that someone who reviews the revisions can readily identify them as multiple versions of a single benchmark.

2. The *<xccdf:Benchmark>* element SHALL have an *@xml:lang* attribute.

3. The *@style* attribute SHOULD have the value "SCAP_1.2".

[14] The schema is posted at http://scap.nist.gov/schema/xccdf/1.2/xccdf-1.2.xsd.

4. The *<xccdf:status>* element SHALL indicate the current status of the benchmark document. The associated text value SHALL be "draft" for documents released in public draft state and "accepted" for documents that have been officially released by an organization. The *@date* attribute SHALL be populated with the date of the status change. Additional *<xccdf:status>* elements MAY be included to indicate historic status transitions.

5. The *<xccdf:version>* element SHALL uniquely identify the particular revision of the benchmark. Also, these revisions SHOULD have version values that indicate the revision sequence, so that the history of changes from the original benchmark can be determined. The *@time* attribute of the *<xccdf:version>* element SHOULD be used for a timestamp of when the benchmark was defined. The *@update* attribute of the *<xccdf:version>* element SHOULD be used for a URI that specifies where updates to the benchmark can be obtained.

6. The *<xccdf:metadata>* element SHALL be provided and SHALL, at minimum, contain the Dublin Core[15] terms from Table 13. Additional Dublin Core terms SHALL follow the required terms within the element sequence.

Table 13 - Use of Dublin Core Terms in XCCDF Metadata

Dublin Core Term	Description of Use
<dc:creator>	The person, organization, and/or service that created the XCCDF XML instance
<dc:publisher>	The person, organization, and/or service that published the XCCDF XML instance
<dc:contributor>	The person, organization, and/or service that contributed to the creation of the XCCDF XML instance
<dc:source>	An identifier that indicates the organizational context of the *<xccdf:Benchmark>* element's *@id* attribute. An organizationally specific URI SHOULD be used.

3.3.4 The <xccdf:Profile> Element

The use of an *<xccdf:Profile>* element SHALL NOT be required. SCAP content commonly includes *<xccdf:Profile>* elements, but they are optional.

3.3.5 Allowed Check System Usage

The following requirements and recommendations apply to the use of the *<xccdf:check>* and *<xccdf:complex-check>* elements:

1. The *<xccdf:check-content>* element SHALL NOT be used to embed check content directly into XCCDF content.

2. At least one *<xccdf:check-content-ref>* element MUST be provided for each *<xccdf:check>*.

3. Use of XCCDF check systems as specified in the *<xccdf:check>* element's *@system* attribute SHALL be restricted as follows:

    a. The following check systems are *supported* by SCAP:

[15] http://dublincore.org/documents/dces/

i. Use of the OVAL check system SHALL be indicated by the *http://oval.mitre.org/XMLSchema/oval-definitions-5* system identifier.

ii. Use of the OCIL check system SHALL be indicated by the *http://scap.nist.gov/schema/ocil/2* system identifier.

b. If a check system is used in XCCDF content that is not *supported* by SCAP, then this content SHALL NOT be considered *well-formed* with regards to SCAP.

If multiple <*xccdf:check-content-ref*> elements occur within an <*xccdf:check*> element, the <*xccdf:check-content-ref*> elements are evaluated in the order they appear. The first resolvable <*xccdf:check-content-ref*> element is used to determine the <*xccdf:Rule*> status. For each <*xccdf:check-content-ref*> element, an implementation attempts to retrieve the document referenced by the element's `@href` attribute. If not resolvable, the next available <*xccdf:check-content-ref*> element is evaluated. If none of the <*xccdf:check-content-ref*> elements are resolvable, then the result of the rule evaluation is the XCCDF "unchecked" status and processing of the <*xccdf:Rule*> ends. The `@href` attribute MUST be resolved in the context of the XML catalog specified as part of the *<ds:component-ref>* that is referencing this benchmark.

3.3.5.1 OVAL <xccdf:check> Usage

References from XCCDF to OVAL definitions SHALL use the form:

```
<xccdf:check-content-ref href="OVAL_Source_URI" [name="OVAL_Definition_Id"]/>
```

The *@href* attribute SHALL reference an OVAL source data stream component. When present, the *@name* attribute SHALL refer to a specific OVAL definition in the designated source data stream component. Use of the *@name* attribute is REQUIRED except for the patches up-to-date rule, as defined in Section 3.3.6.4.

In the previous example, the *<xccdf:check-content-ref>* element's *@href* attribute refers to an OVAL source data stream component containing one or more OVAL patch class definitions. This *<xccdf:check-content-ref>* is equivalent to *referencing* a virtual OVAL definition of the form:

```
<oval_definitions xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5">
   <definitions>
      <definition id="identifier of patch definition" version="0" class="patch">
         …
         <criteria>
            <extend_definition definition_ref="identifier of patch definition 1"/>
            …
            <extend_definition definition_ref="identifier of patch definition N"/>
         </criteria>
      </definition>
   </definitions>
</oval_definitions>
```

where the extended definitions are the individual patch class definitions defined in the OVAL source data stream component.

See Section 4.5.2 for additional information on mapping OVAL results to XCCDF results.

3.3.5.2 <xccdf:Value> and OVAL Variable Dependencies

One or more *<xccdf:check-export>* elements MAY be used to define the binding of *<xccdf:Value>* elements to OVAL variables. The format of the *<xccdf:check-export>* element is:

```
<xccdf:check-export value-id="XCCDF_Value_id"
export-name="OVAL_External_Variable_id"/>
```

The following <xccdf:check> element example demonstrates the use of this convention:

```
<xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
   <xccdf:check-export value-id="NoSlowLink_var"
   export-name="oval:gov.nist.fdcc.xp:var:66711"/>
   <xccdf:check-export value-id="NoBackgroundPolicy_var"
   export-name="oval:gov.nist.fdcc.xp:var:66712"/>
   <xccdf:check-export value-id="NoGPOListChanges_var"
   export-name="oval:gov.nist.fdcc.xp:var:66713"/>
   <xccdf:check-content-ref href="fdcc-winxp-oval.xml"
   name="oval:gov.nist.fdcc.xp:def:6671"/>
</xccdf:check>
```

The type and value binding of the specified *<xccdf:Value>* is constrained to match that lexical representation of the indicated OVAL Variable Data Type. Table 14 summarizes the constraints regarding data type usage. Additional information regarding OVAL and XCCDF data types can be found in the OVAL Common Schema documentation[16] and the XCCDF specification [XCCDF].

Table 14 - XCCDF-OVAL Data Export Matching Constraints

OVAL Data Type	Matching XCCDF Data Type
int	number
float	number
boolean	boolean
string, evr_string, version, ios_version, fileset_revision, binary	string

3.3.5.3 OCIL <xccdf:check> Usage

When referencing OCIL questionnaires as checks, XCCDF content SHALL follow all requirements defined in Appendix B of NIST Interagency Report (IR) 7692, *Specifications for the Open Checklist Interactive Language (OCIL) Version 2.0* [OCIL].

[16] http://oval.mitre.org/language/download/schema/version5.4/ovaldefinition/documentation/oval-common-schema.html#DatatypeEnumeration and http://oval.mitre.org/language/download/schema/version5.3/ovaldefinition/documentation/oval-definitions-schema.pdf

3.3.6 The <xccdf:Rule> Element

The following requirements and recommendations apply to the *<xccdf:Rule>* element.

3.3.6.1 Identifier Use

Each *<xccdf:Rule>* element SHALL include an *<xccdf:ident>* element containing a CVE, CCE, or CPE identifier reference if an appropriate reference exists. If the rule references an OVAL definition, then *<xccdf:ident>* element content SHALL match the corresponding CVE, CCE, or CPE identifier found in the associated OVAL definition(s) if an appropriate identifier exists.

When referencing a CVE, CCE, or CPE identifier, an *<xccdf:Rule>* element MUST be consistent with one of the rows in Table 15. Based on the purpose of the *<xccdf:Rule>* element, the *<xccdf:Rule>* SHALL define its *<xccdf:ident>* element's *@system* attribute using the corresponding value from Table 15. Also, if the *<xccdf:Rule>* element references an OVAL definition, it SHALL reference an OVAL definition of the specified class that SHALL reference an identifier of the specified type.

Table 15 – Identifier Use for <xccdf:Rule> Elements

Purpose of the <xccdf:Rule>	OVAL Definition Class	Identifier Type	Value for <xccdf:ident> @system attribute
Check compliance with a configuration setting	compliance	CCE	http://cce.mitre.org
Perform a software inventory check	inventory	CPE	http://cpe.mitre.org
Detect the presence of a software flaw vulnerability	vulnerability	CVE	http://cve.mitre.org

Here is a partial example of a rule intended to check compliance with a configuration setting:

```
<xccdf:Rule id="AuditAccountLogonEvents">
    …
    <xccdf:ident system="http://cce.mitre.org">CCE-3867-0</xccdf:ident>
    …
</xccdf:Rule>
```

See Section 4.5.1 for information on the meaning of a "pass/fail" rule result relating to each of the identifier types in Table 15.

An *<xccdf:ident>* element referencing a CVE, CCE, or CPE identifier SHALL be ordered before other *<xccdf:ident>* elements referencing non-SCAP identifiers. Identifiers from previous revisions of CCE or CPE MAY also be specified following the SCAP identifiers.

3.3.6.2 OVAL Definition References

If an *<xccdf:Rule>* element references a specific OVAL definition, then:

1. The referenced OVAL definition MUST have its *@class* attribute defined as "compliance" if it represents a check for the value of a specific configuration setting.

2. The referenced OVAL definition MUST have its *@class* attribute defined as "vulnerability" if it represents a check for the presence of a particular software flaw vulnerability.

3. The referenced OVAL definition MUST have its *@class* attribute defined as "patch" if it represents a check for the presence of a discrete patch.

4. The referenced OVAL definition MUST have its *@class* attribute defined as "inventory" if it represents a check for the presence of a product of interest.

3.3.6.3 OCIL Questionnaire References

An XCCDF rule MAY reference an OCIL questionnaire. This SHOULD be done only for cases where OVAL cannot perform the check.

3.3.6.4 Use of a Patches Up-To-Date Rule

An OVAL instance document MAY be used to represent a series of checks to verify that patches have been installed. Historically, an XCCDF convention has been used to identify such a reference. An XCCDF benchmark MAY include a patches up-to-date rule that references an OVAL source data stream component. When implementing a patches up-to-date XCCDF rule, the following approach SHALL be used:

1. The source data stream MUST include an OVAL source data stream component with one or more OVAL patch class definitions.

2. The *<xccdf:Rule>* element that references an OVAL source data stream component SHALL have the *@id* attribute value of "*security_patches_up_to_date*".

3. A single *<xccdf:check>* element SHALL be provided for the *<xccdf:Rule>* with a *@system* attribute value of "*http://oval.mitre.org/XMLSchema/oval-definitions-5*".

4. Each *<xccdf:check-content-ref>* element SHALL have an *@href* attribute referencing a valid SCAP *<oval-def:oval_definitions>* document instance with the *@name* attribute omitted.

5. The *@multi-check* attribute of the *<xccdf:Rule>* element SHOULD be set to "true". This causes a separate *<xccdf:rule-result>* to be generated for each check. See Section 4.5.2 for more information.

For example:

```
<xccdf:Rule id="security_patches_up_to_date" selected="false">
   <xccdf:title>Security Patches Up-To-Date</xccdf:title>
   <xccdf:description>Keep systems up to current patch levels
   </xccdf:description>
   <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
      <xccdf:check-content-ref href="scap-win2000-patches.xml"/>
   </xccdf:check>
</xccdf:Rule>
```

3.3.6.5 CVSS and CCSS Scores

SCAP 1.0 required the inclusion of static CVSS scores in XCCDF vulnerability-related rules. However, CVSS base scores sometimes change over time, such as when more information is available about a

particular vulnerability, and CVSS temporal and environmental scores are intended to change to reflect current threats, security controls, and other factors. Current CVSS scores acquired dynamically, such as from a data feed, SHOULD be used in place of static CVSS scores in the `@weight` attribute within XCCDF vulnerability-related rules. Section 3.9 contains additional requirements for CVSS usage.

CCSS scores are more stable than CVSS scores, but they still may change over time. Accordingly, current CCSS scores acquired dynamically, such as from a data feed, MAY be used in place of static CCSS scores in the `@weight` attribute within XCCDF configuration setting-related rules. Section 3.10 contains additional requirements for CCSS usage.

3.3.7 The <xccdf:Value> Element

The use of the `<xccdf:source>`, `<xccdf:complex-value>`, and `<xccdf:complex-default>` elements within the `<xccdf:Value>` element SHALL NOT be allowed. Within the `<xccdf:choices>` element of the `<xccdf:Value>` element, the use of the `<xccdf:complex-choice>` element SHALL NOT be allowed.

3.4 Open Vulnerability and Assessment Language (OVAL)

While the default version[17] of OVAL used in SCAP 1.2 SHALL be OVAL version 5.10, SCAP content SHOULD utilize the earliest SCAP-supported version of OVAL (5.3 at minimum) that includes all required tests and is necessary to properly address the content's purpose or use case. This approach, often referred to as the "least version principle", allows for SCAP content to remain viable over a longer period of time by allowing for the broadest support within products, while reducing the content maintenance burden that would be required to maintain revisions of content for multiple specification versions.

All of the OVAL content MUST contain an *<oval:generator>* element. The version of any particular document instance SHALL be specified using the *<oval:schema_version>* content element of the *<oval:generator>* as in this example:

```
<oval:generator>
  <oval:product_name>The OVAL Repository</oval:product_name>
  <oval:schema_version>5.10</oval:schema_version>
</oval:generator>
```

The version of an *<oval-var:oval_variables>* document SHALL be the same as that of the *<oval-def:oval_definitions>* document whose external variables are bound by the variables document.

The following requirements apply to particular classes of OVAL definitions:

1. For compliance class definitions:

    a. If an OVAL compliance class definition maps to one or more CCE identifiers, the definition SHOULD include *<oval-def:reference>* elements that reference those identifiers using the following format:

    ```
    <oval-def:reference source="http://cce.mitre.org"
    ref_id="CCE_identifier"/>
    ```

[17] The OVAL language versioning methodology is available here: http://oval.mitre.org/language/about/versioning.html

The source attribute SHALL be defined using either "*http://cce.mitre.org"* (preferred method) or "CCE".

b. Definitions that are directly or indirectly extended SHALL be limited to inventory and compliance classes.

2. For inventory class definitions:

a. If an OVAL inventory class definition maps to one or more CPE identifiers, the definition SHOULD include *<oval-def:reference>* elements that reference those identifiers using the following format:

```
<oval-def:reference source="http://cpe.mitre.org"
ref_id="CPE_identifier"/>
```

The source attribute SHALL be defined using either "*http://cpe.mitre.org"* (preferred method) or "CPE".

b. Definitions that are directly or indirectly extended SHALL be limited to the inventory class.

3. For patch class definitions:

a. If an OVAL patch class definition maps to one or more CVE identifiers, the definition MAY include *<oval-def:reference>* elements that reference those identifiers using the following format:

```
<oval-def:reference source="http://cve.mitre.org"
ref_id="CVE_identifier"/>
```

The source attribute SHALL be defined using either "*http://cve.mitre.org"* (preferred method) or "CVE".

b. If an OVAL patch class definition is associated with a source specific identifier (for example, Knowledge Base numbers for Microsoft patches), these identifiers SHOULD be included in *<oval-def:reference>* elements contained by the definition. For example:

```
<oval-def:reference source="www.microsoft.com/Patch"
ref_id="KB912919"/>
```

c. Definitions that are directly or indirectly extended SHALL be limited to inventory and patch classes.

4. For vulnerability class definitions:

a. If an OVAL vulnerability class definition maps to one or more CVE identifiers, the definition SHOULD include *<oval-def:reference>* elements that reference those identifiers using the following format:

```
<oval-def:reference source="http://cve.mitre.org"
ref_id="CVE_identifier"/>
```

The source attribute SHALL be defined using either "*http://cve.mitre.org"* (preferred method) or "CVE".

b. Definitions that are directly or indirectly extended SHALL be limited to inventory and vulnerability classes.

All OVAL components SHALL validate against the corresponding Schematron rules for OVAL available at http://scap.nist.gov/revision/1.2/#schematron. In most cases, the OVAL Schematron posted at the SCAP website is the same Schematron that was released with the official OVAL release. Minor modifications to the Schematron files are made when issues in the OVAL Schematron files are discovered. All modifications are documented on the SCAP website.

3.5 Open Checklist Interactive Language (OCIL)

OCIL content SHOULD be used for checking rules that cannot be fully automated with OVAL.[18] For example, a particular software product may not have an application programming interface (API) that supports OVAL use. Another example is performing a check that requires user interaction, such as asking the user to look up information within a management console or to report a serial number affixed to a computing device. OCIL can also be used to collect a user's own information, such as whether the user participated in a recent security training session.

3.6 Common Platform Enumeration (CPE)

The Official CPE Dictionary data feed[19] MAY be used by SCAP components to reference CPE names. In cases where the Official CPE Dictionary is impractical for use, a subset of the dictionary MAY be used. In order to create the reduced official dictionary, every CPE in an *<xccdf:platform>* or *<cpe-lang:fact-ref>* element in every benchmark in the data stream referencing this CPE dictionary MUST be matched against every entry in the Official CPE Dictionary using the CPE name matching algorithm [CPE-M]. All CPEs matched in the official dictionary with a result of EQUAL or SUPERSET MUST be included in the resulting dictionary. One or more third-party dictionaries MAY be included in a data stream as well. The third-party dictionary being utilized SHOULD follow the requirements of the CPE Dictionary specification [CPE-D]. If including a third-party dictionary is impractical, a subset of the dictionary MAY be included. The reduced dictionary MUST be created using the same procedure outlined for creating a subset of the official dictionary. In all cases, the dictionary component MAY be remote to the data stream collection.

[CPE-D] provides the defining structure of a CPE dictionary. A *<cpe_dict:cpe-item>* MAY contain one or more *<cpe-dict:check>* elements that reference OVAL inventory class definitions using the following format:

```
<cpe_dict:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5"
      [href="oval_URL"]>oval_inventory_definition_id</cpe_dict:check>
```

For example:

```
<cpe_dict:cpe-list xmlns="http://cpe.mitre.org/dictionary/2.0"
          xmlns:cpe_dict="http://cpe.mitre.org/dictionary/2.0">
   <cpe_dict:cpe-item name="cpe:2,3:o:microsoft:windows_2003:*.*.*.*.*.*.*.*">
      <cpe_dict:title>Microsoft Windows Server 2003</title>
      <cpe_dict:check
          system=http://oval.mitre.org/XMLSchema/oval-definitions-5
          href="example-winsvr2003-oval.xml"> oval:org.mitre.oval:def:128
```

[18] The OCIL specification is available at http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7692.
[19] The Official CPE Dictionary is located at http://nvd.nist.gov/cpe.cfm.

```
      </cpe_dict:check>
   </cpe_dict:cpe-item>
</cpe_dict:cpe-list>
```

The referenced OVAL inventory class definition SHALL specify the technical procedure for determining whether or not a specific target asset is an instance of the CPE name specified by the *<cpe_dict:cpe-item>* element. This usage is encouraged for a CPE dictionary source data stream component.

When creating a subset of the Official CPE Dictionary or a third-party dictionary, a *<cpe_dict:check>* element on an entry MAY be added or modified if the existing check does not provide satisfactory content to test the presence of the CPE name.

If a *<cpe_dict:cpe-item>* contained in a CPE dictionary data stream component references an OVAL inventory class definition, then that definition SHALL be resolved by an *@href* attribute referencing an OVAL source data stream component in the same data stream.

3.7 Common Configuration Enumeration (CCE)

To maintain consistency and accuracy, SCAP content referencing a configuration setting SHALL use the official CCE identifier if a CCE entry for a particular configuration setting exists in the Official CCE Dictionary. If no CCE exists for the configuration setting of interest, the content author SHOULD seek to have a CCE identifier issued for the configuration setting. See the OVAL compliance class definition requirements in Section 3.4 and the <xccdf:ident> requirements in Section 3.3.6.1for additional requirements involving CCE identifier references.

The MITRE Corporation maintains the current official CCE list at http://cce.mitre.org/lists/cce_list.html and new CCEs can be requested from the CCE Content Team at http://cce.mitre.org/lists/creation_process.html.

Use of an official, dynamic data feed is preferred to static coding of values in SCAP data sources. The NVD provides a data feed[20] that correlates CCE identifiers with the control identifiers described in NIST SP 800-53. Embedding control identifiers within SCAP content is strongly discouraged due to the maintenance burden that it imposes on content maintainers when the control identifiers are revised.

3.8 Common Vulnerabilities and Exposures (CVE)

CVE references in SCAP content MAY include both “candidate” and “entry” status identifiers. The use of deprecated CVE identifiers SHALL NOT be allowed.

If a CVE identifier exists for a particular vulnerability, the official CVE identifier SHALL be used. If no CVE exists for the software flaw, an alternate identifier MAY be used, but the user SHOULD seek to have a CVE identifier issued for the vulnerability. The process for submitting unpublished vulnerabilities and obtaining CVE identifiers is available from The MITRE Corporation via http://cve.mitre.org/cve/obtain_id.html.

NIST provides a CVE data feed to support dynamic and current vulnerability information and associated metadata (e.g., CVSS values). The current schema is available at http://nvd.nist.gov/download.cfm.

[20] http://nvd.nist.gov/cce.cfm

3.9 Common Vulnerability Scoring System (CVSS)

The NIST CVE data feed, discussed in Section 3.8, is one source of CVSS base score and vector data that MAY be used by products to support additional use cases built on SCAP usage. In support of these additional use cases, CVSS base scores and vectors from this data feed MAY be used by products along with temporal and environmental scores and vectors from other sources.

Additional information on CVSS use is available in NIST IR 7435, *The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems* (http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7435).

3.10 Common Configuration Scoring System (CCSS)

CCSS base, temporal, and environmental scores and vectors MAY be used by products. Additional information on CCSS use is available in NIST IR 7502, *The Common Configuration Scoring System (CCSS): Metrics for Software Security Configuration Vulnerabilities* (http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7502).

3.11 XML Digital Signature

Digitally signing source data stream content is important to ensuring the integrity and trustworthiness of legitimate content, while preventing rogue content from being executed. Leveraging the Trust Model for Security Automation Data (TMSAD) specification [TMSAD] for SCAP can improve the legitimacy of authoritative content and create a more secure environment. As such, content authors MAY digitally sign source content following the guidelines in [TMSAD], along with the following requirements.

One or more XML digital signatures MAY be included as the last elements in the SCAP source data stream collection root element. Each signature MUST be represented as a `<dsig:Signature>` and follow the W3C recommendation [DSIG]. Each `<dsig:Signature>` element MUST sign only one data stream.

The `<dsig:Signature>` element MUST follow the recommendations in [TMSAD] and these additional requirements:

1. A `<dsig:Manifest>` MUST be included in the `<dsig:Signature>` as a `<dsig:Object>`. The `<dsig:Manifest>` MUST have a `<dsig:Reference>` for each local component referenced by the data stream being signed. External components MAY be omitted from the `<dsig:Manifest>`. Each `<dsig:Reference>` to a local component MUST point to the component being signed by identifying the component in the `@URI` attribute using "#" + `@Id` of the component.

2. A `<dsig:SignatureProperties>` MUST be included in the `<dsig:Signature>` as a `<dsig:Object>`. The `<dsig:SignatureProperties>` MUST be populated in accordance with the guidelines in [TMSAD].

3. The first `<dsig:Reference>` in `<dsig:Signature>` MUST be to the `<ds:data-stream>` element being signed. The `<ds:data-stream>` MUST be referenced in the `@URI` attribute using "#" + `@Id` of the `<ds:data-stream>`.

4. The second *<dsig:Reference>* in *<dsig:Signature>* MUST be to the *<dsig:SignatureProperties>* captured in a *<dsig:Object>* within the *<dsig:Signature>*. The *<dsig:SignatureProperties>* MUST be referenced in the *@URI* attribute using "#" + *@Id* of the*<dsig:SignatureProperties>*.

5. The third *<dsig:Reference>* MUST be to the *<dsig:Manifest>* captured in a *<dsig:Object>* with the *<dsig:Signature>*. The *<dsig:Manifest>* MUST be referenced in the *@URI* attribute using "#" + *@Id* attribute of the *<dsig:Manifest>*.

6. *<dsig:Reference>* elements on the *<dsig:Manifest>* SHOULD be in the same order as the *<ds:component-ref>* elements on the data stream being signed.

7. Key information SHOULD be provided on the *<dsig:Signature>* in accordance with [TMSAD].

4. SCAP Processing Requirements and Recommendations

This section defines the processing requirements that SCAP content consumers MUST follow in order to correctly process SCAP 1.2 content. This section also provides recommendations that are not mandatory; organizations are encouraged to adopt them to promote stronger interoperability and greater consistency. The topics covered in the first part of this section are legacy support, source data streams, and XCCDF processing. The end of the section covers result-related topics: SCAP result data streams, XCCDF results, OVAL results, OCIL results, and result data stream signing.

4.1 Legacy Support

Content consumers supporting SCAP 1.2 SHALL process SCAP 1.2 content, SHALL process SCAP 1.1 content, and SHALL process SCAP 1.0 content. Content consumers that process legacy SCAP content, content defined using an SCAP version prior to 1.2, SHALL process it as defined under the corresponding version of NIST SP 800-126 (for SCAP 1.1, revision 1; for SCAP 1.0, the original version).[21] Content consumers that process legacy SCAP content MUST be capable of outputting results in the same SCAP version as the source content, and MAY convert the legacy SCAP results into a newer SCAP results version.

Content consumers supporting OVAL SHALL support OVAL definition documents written against OVAL versions 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, and 5.10.

Within the OVAL language, constructs may be deprecated.[22] Deprecated constructs MUST be handled properly during OVAL definition evaluation. Similar to the requirement to support previous minor versions of OVAL, this requirement ensures that legacy content that made use of these deprecated constructs continues to be supported in SCAP.

4.2 Source Data Streams

Content consumers SHALL be capable of validating the content against the appropriate schemas and Schematron stylesheets, detecting and reporting errors, and failing gracefully if there are errors. The relevant XML schemas are located at http://scap.nist.gov/revision/1.2/#schema. The locations of the relevant Schematron rule sets are specified in Sections 3.1 and 3.4, as well as directly in the OCIL specification discussed in Section 3.5.

Content consumers SHOULD validate the XML digital signatures if they exist in the content. Validating a signature includes confirming that the signature value is valid, all of the reference hashes in the signature and manifest are correct, and the public key used to verify the signature is from a trusted source. A data stream with a digital signature that does not validate SHOULD NOT be evaluated by a content consumer.

If a content consumer imports a data stream that references an extended component that it does not recognize, it SHALL issue an error indicating the reason for the error.

If more than one data stream is specified on the data stream collection, the ID of the data stream to execute must be indicated to the content consumer, and the content consumer MUST use the specified data stream. If more than one XCCDF benchmark is referenced by a data stream, the ID of the

[21] http://csrc.nist.gov/publications/PubsSPs.html#800-126

[22] The OVAL Language Deprecation policy is available here: http://oval.mitre.org/language/about/deprecation.html

benchmark to execute MUST be indicated to the content consumer, and the content consumer MUST process the indicated benchmark.

4.3 XCCDF Processing

The following requirements and recommendations pertain to content consumers processing XCCDF content.

4.3.1 CPE Applicability Processing

When evaluating an *<xccdf:platform>* element in XCCDF content, it is necessary to evaluate machine state to determine the presence of a referenced CPE on the machine. CPEs referenced in an *<xccdf:platform>* element directly or by a *<cpe-lang:fact-ref>* contained within a referenced *<cpe-lang:platform-specification>* element SHALL be evaluated as follows:

1. The CPE SHALL be matched against all CPEs in all of the dictionaries referenced by the data stream. All CPEs that return an EQUAL or SUPERSET result as defined in CPE Matching [CPE-M] SHALL be used in evaluating the *<xccdf:platform>* or *<cpe-lang:fact-ref>*.

2. The *<cpe_dict:check>* element data associated with the found *<cpe_dict:cpe-item>* elements SHALL be evaluated against the target using the referenced OVAL inventory class definition when the CPEs installed on the target are not known prior to the evaluation.

3. The result of evaluation SHALL be handled according to Section 4.5.2 of this document, with a result of "pass" indicating that the CPE name was found on the machine. In cases where the list of CPEs installed on the target is already known, the content consumer MAY do name matching between the source CPE dictionary names and the target CPE list instead of scanning the target. Results of name matching SHALL be handled in a manner consistent with when an evaluation is executed to determine existence.

4.3.2 Check System Usage

In XCCDF content, if multiple *<xccdf:check-content-ref>* elements are provided, then the following evaluation method SHALL be performed:

1. Evaluate each *<xccdf:check-content-ref>* element in the order that it appears in the *<xccdf:check>* element. The first resolvable *<xccdf:check-content-ref>* element SHALL be used to determine the *<xccdf:Rule>* status.

2. For each *<xccdf:check-content-ref>* element, a content consumer will attempt to retrieve the document referenced by the *@href* attribute within the context of the XML Catalog specified as part of the *<ds:component-ref>* used to reference this benchmark. If not resolvable, the next available *<xccdf:check-content-ref>* element SHALL be evaluated. If none of the *<xccdf:check-content-ref>* elements are resolvable, then the result of the rule evaluation SHALL be the XCCDF "unchecked" status and processing of the *<xccdf:Rule>* SHALL end.

3. Once a resolvable *<xccdf:check-content-ref>* element is found, then check system processing SHALL proceed. When evaluating a rule, an *<xccdf:rule-*

result/xccdf:message> with the *@severity* attribute value of "info" SHALL be generated, indicating the <*xccdf:check-content-ref*> *@href* and *@name*, if provided.

Use of XCCDF check systems as specified in the *<xccdf:check>* element's *@system* attribute SHALL be restricted as follows:

1. Content consumers SHALL implement the SCAP-conformant check systems that are required for the SCAP use case or use cases that the content consumers support. The SCAP-conformant check systems are:

   i. OVAL check system. Use of the OVAL check system SHALL be indicated by the *http://oval.mitre.org/XMLSchema/oval-definitions-5* system identifier.

   ii. OCIL check system. Use of the OCIL check system SHALL be indicated by the *http://scap.nist.gov/schema/ocil/2* system identifier.

2. Content consumers MAY implement check systems that are not supported by SCAP.

An *<xccdf:check-content-ref>* element may omit the *@name* attribute only for a patches up-to-date rule (see Section 3.3.6.4). When processing a patches-up-to-date rule, only OVAL patch class definitions SHALL be evaluated; all other classes of definitions (e.g., inventory class definitions) SHALL NOT be evaluated.

4.4 SCAP Result Data Streams

An SCAP result data stream contains the results of the evaluation of one or more SCAP source data streams by an SCAP content consumer. Correlation and aggregation products such as security awareness incident response tools may consume properly formatted SCAP result data streams to support organizational reporting requirements.

A result data stream SHALL conform to the [ARF] specification. The following sections outline the details of the ARF report. In all situations, one or more component results (e.g., XCCDF, check results), the target asset, and/or the SCAP source data stream collection represented as a report request in ARF MAY be represented either as a local component in the ARF or as a remote resource, leveraging the remote resource capability built into ARF.

4.4.1 The Component Reports

The ARF report MUST contain a report object for each benchmark and check component executed when a source data stream runs against a target. Specifically, there SHALL be an OVAL result report for each OVAL component that was executed during the run, and there SHALL be an OCIL result report for each OCIL component that was executed during the run. If an XCCDF benchmark was executed during the run, then there SHALL be a XCCDF result report for that as well. Each component result MUST be captured as a separate report object in the ARF report and each component report SHALL use the element specified in Table 16 as its root element.

Table 16 - SCAP Result Data Stream Document Elements

Component	Document Element
XCCDF Benchmark	`<xccdf:Benchmark>` or `<xccdf:TestResults>`
OVAL	`<oval-def:oval_definitions>`
OCIL Questionnaire	`<ocil:ocil>`

Each SCAP result data stream component SHOULD NOT use any constructs that are deprecated in its associated specification. Validation of each component SHALL be done in accordance with the portions of this document that define requirements for the component. See Section 3.2 for more information on the SCAP Content Validation Tool, which can help validate the correctness of SCAP result data streams.

4.4.2 The Target Identification

The target asset MUST be represented in the ARF report using the "assets" part of ARF. The asset identification element populated about a target asset SHOULD include the fields specified in Table 17, where applicable.

Table 17 – Asset Identification Fields to Populate

Field	Location within Asset Identification Computing Device
Ethernet media access control address	connections/connection/mac-address
Internet Protocol version 4 address	connections/connection/ip-address/ip-v4
Internet Protocol version 6 address	connections/connection/ip-address/ip-v6
Host name of the asset, if assigned	hostname
Fully qualified domain name	fqdn

Additional identification information MAY be captured in the asset identification element (asset tag, system GUID, etc.) The guidelines specified in [AI] MUST be followed when populating the asset identification information.

4.4.3 The Source Data Stream

The source data stream collection that was used to generate the results against the target SHOULD be included in the ARF report as a report request.

4.4.4 The Relationships

Table 18 outlines the relationships that MUST be specified in the ARF report if the stated condition is satisfied.

Table 18 – ARF Relationships

Relationship	Condition	Cardinality	Definition	Subject	Object
arf-rel:isAbout	None	One for each component report	Each report is reporting about the asset	Component report	Target asset
scap-rel:checkContext	Benchmark report exists	One for each check component report (OVAL or OCIL)	Each check report is reporting in the context of the benchmark report	Check component report	Benchmark component report
scap-rel:fromSource	Report request exists	One for each component report	Each component report was generated from the SCAP source content	Component report	Report request

Figure 3 gives an example of how the resulting ARF report would look.

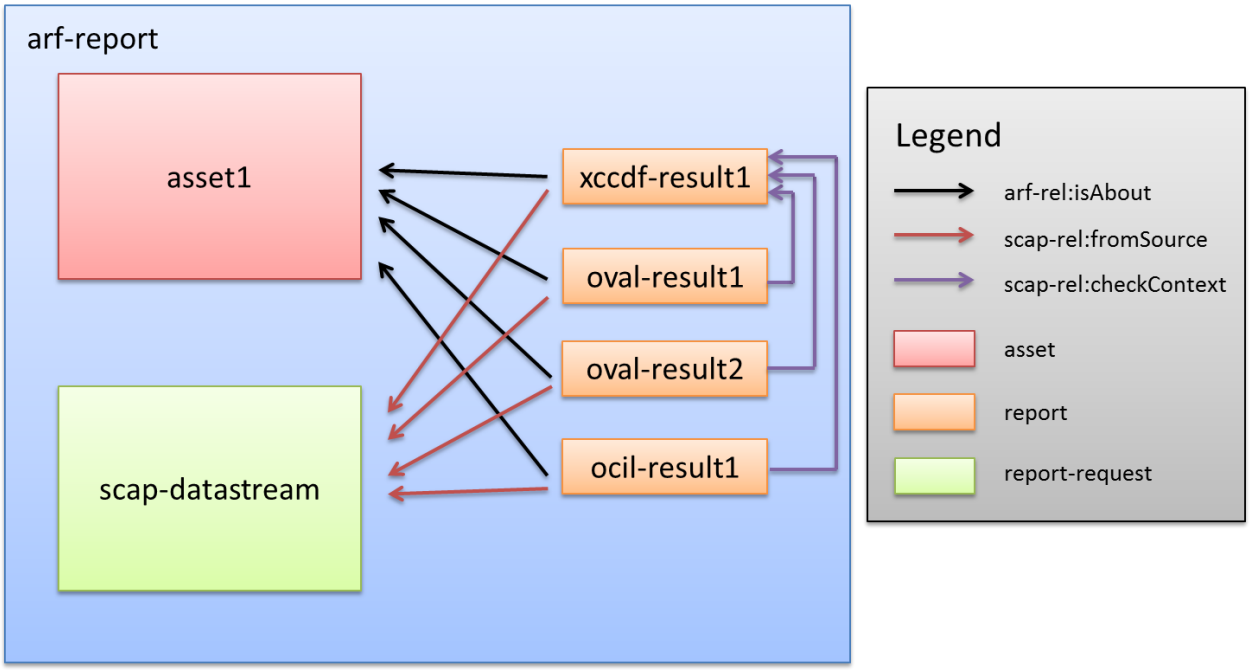

Figure 3 – Sample ARF Report Structure

4.5 XCCDF Results

Each XCCDF result data stream component SHALL comply with the XCCDF Results schema.

XCCDF test results SHALL be documented as the contents of an *<xccdf:TestResult>* element. *<xccdf:benchmark>* elements SHALL be ignored in *<xccdf:TestResult>* elements that are embedded as a child-element of an *<xccdf:Benchmark>* root element.

To be considered valid SCAP result content, the following conditions SHALL be met:

1. One or more *<xccdf:organization>* elements SHALL be provided to indicate the organizational units responsible for applying the checklist.

2. The *@start-time* and *@end-time* attributes SHALL be provided to indicate when the scan started and completed, respectively.

3. The *@test-system* attribute SHALL be provided with a CPE name value indicating the product that evaluated the checklist.

4. If the *<xccdf:TestResult>* is the root XCCDF element, the *<xccdf:benchmark>* element's *@href* attribute SHALL be an absolute URL, NOT a relative URL.

5. Regarding the definition and use of *<xccdf:Profile>* elements:

   a. If no *<xccdf:Profile>* was selected, then the *<xccdf:Profile>* SHALL be omitted.

   b. When using a profile during the processing of XCCDF content, the test results SHALL embed an *<xccdf:profile>* element that contains the name of the utilized profile.

   c. Reported *<xccdf:set-value>* elements SHALL include all those values that are exported by the reported rules. The specific settings are those determined by the reported Profile.

6. Reported *<xccdf:rule-result>* elements SHALL include all rules selected during processing.

7. The *<xccdf:identity>* element SHALL identify the security principal used to access rule evaluation on the target(s). This will include the identity name or username used to perform the scan.

8. Each IP address associated with the *<xccdf:target>* SHALL be enumerated using the *<xccdf:target-address>* element.

9. In *<xccdf:target-facts>*, an *<xccdf:target-id-ref>* SHALL be specified with a *@system* attribute of "http://scap.nist.gov/schema/asset-identification/1.1", an *@href* value of "", and a *@name* value of the ID of the asset identification element in the ARF that this *<xccdf:TestResult>* is about.

10. The *<xccdf:rule-result>* elements SHALL report the result of the application of each selected rule against all specified checks. The *<xccdf:check/xccdf:check-content-ref>* element SHALL record the reference to the check system specific result component report ID and check name within the result file using the *@href* and *@name* attributes, respectively. This approach provides traceability between XCCDF and check results.

11. Where applicable to the target system, each of the following *<xccdf:fact>* elements SHALL be provided:

Table 19 - XCCDF Fact Descriptions

XCCDF Fact	Description of Use
urn:scap:fact:asset:identifier:mac	Ethernet media access control address
urn:scap:fact:asset:identifier:ipv4	Internet Protocol version 4 address
urn:scap:fact:asset:identifier:ipv6	Internet Protocol version 6 address
urn:scap:fact:asset:identifier:host_name	Host name of the asset, if assigned
urn:scap:fact:asset:identifier:fqdn	Fully qualified domain name

XCCDF Fact	Description of Use
urn:scap:fact:asset:identifier:ein	Equipment identification number or other inventory tag number
urn:scap:fact:asset:identifier:guid	Globally unique identifier for the asset, if assigned
urn:scap:fact:asset:environmental_information:owning_organization	Organization that tracks the asset on its inventory
urn:scap:fact:asset:environmental_information:current_region	Geographic region where the asset is located
urn:scap:fact:asset:environmental_information:administration_unit	Name of the organization that does system administration for the asset

4.5.1 Assigning Identifiers to Rule Results

The *<xccdf:rule-result>* element provides data indicating the result of assessing a system using the identified *<xccdf:Rule>* element. If the target *<xccdf:Rule>* identified by the <xccdf:*rule-result*> *@idref* attribute has one or more *<xccdf:ident>* elements with a *@system* attribute value listed in Section 3.3.6.1, item 2, then each *<xccdf:ident>* element SHALL also appear within the *<xccdf:rule-result>* element.

Here is an example for a CVE entry:

```
<xccdf:rule-result idref="java-upgrade-278" weight="10.0">
   <xccdf:result>pass</xccdf:result>
   …
   <xccdf:ident system="http://cve.mitre.org">CVE-2006-0614</xccdf:ident>
   …
</xccdf:rule-result>
```

An *<xccdf:rule-result>* of "pass" SHALL indicate that the check content evaluated within the rule complied with one of the following:

- For a CVE entry, the target platform satisfies all the conditions of the XCCDF rule and is unaffected by the vulnerability or exposure referenced by the CVE.
- For a CCE entry, the target platform complies with the configuration setting guidance expressed in the XCCDF rule.
- For a CPE entry, the target platform was identified on the system.

4.5.2 Mapping OVAL Results to XCCDF Results

When evaluating an *<xccdf:Rule>* element that references an OVAL definition, the *<xccdf:rule-result>* element SHALL be used to capture the result of this evaluation. This result SHALL be determined by evaluating the referenced OVAL definition on a target host. The *<xccdf:result>* value recorded SHALL be mapped from the OVAL definition result produced during evaluation. While the OVAL specification permits limiting result status reporting, SCAP-conformant content SHALL include full status reporting, including Error, Unknown, Not Applicable, Not Evaluated, True, and False.

Content consumers that generate XCCDF *<xccdf:rule-result>* elements SHALL apply the mapping illustrated in Table 20 when deriving *<xccdf:Rule>* results from OVAL definition

processing. The corresponding *<xccdf:rule-result/xccdf:result>* value SHALL be recorded based on the *@class* of the OVAL definition where applicable.

Table 20 - Deriving XCCDF Rule Results from OVAL Definition Results

<table>
<tr><th colspan="2">OVAL Definition Result</th><th>XCCDF Rule Result</th></tr>
<tr><td colspan="2">error</td><td>error</td></tr>
<tr><td colspan="2">unknown</td><td>unknown</td></tr>
<tr><td colspan="2">not applicable</td><td>notapplicable</td></tr>
<tr><td colspan="2">not evaluated</td><td>notchecked</td></tr>
<tr><td>Definition Class</td><td>Definition Result</td><td rowspan="5">Pass</td></tr>
<tr><td>compliance</td><td>true</td></tr>
<tr><td>vulnerability</td><td>false</td></tr>
<tr><td>inventory</td><td>true</td></tr>
<tr><td>patch</td><td>false</td></tr>
<tr><td>Definition Class</td><td>Definition Result</td><td rowspan="5">Fail</td></tr>
<tr><td>compliance</td><td>false</td></tr>
<tr><td>vulnerability</td><td>true</td></tr>
<tr><td>inventory</td><td>false</td></tr>
<tr><td>patch</td><td>true</td></tr>
</table>

The mappings in Table 20 are specific to each OVAL definition class. For example, if an OVAL compliance class definition is processed and the XCCDF returns a result of "true", the content consumer is conveying the fact that the system was found to be compliant with that check and therefore returns a "pass" result. A similar definition for a vulnerable condition will return results of "false" if that vulnerability was not found on the examined devices, resulting in a "pass" from the XCCDF rule.

If the *<xccdf:Rule>* under evaluation has an *<xccdf:check-content-ref>* element with the *@name* attribute omitted and a *@multi-check* attribute set to "true", then the result of each evaluated OVAL definition SHALL be recorded as a separate *<xccdf:rule-result>*. This will commonly occur for a *"security_patches_up_to_date"* check, as defined in Section 3.3.6.4. In this case the *<xccdf:rule-result/xccdf:check-content-ref>* SHALL identify the specific check result of each evaluated OVAL definition using the *@href* and *@name* attributes as described in Section 4.5, item 10.

4.5.3 Use of the FDCC Reporting Format

In SCAP 1.0 and 1.1, SCAP-conformant products were required to support the FDCC XCCDF results format.[23] These requirements have been integrated into SCAP 1.2. To produce an SCAP 1.2 result data stream that is conformant with the FDCC XCCDF results reporting format, the XCCDF result component SHALL use *<xccdf:TestResult>* as the top-level element. Additionally, an organization SHALL document deviations using the *<xccdf:override>* element for any *<xccdf:rule-result>* element with an *<xccdf:result>* element value that is not "Pass".

FDCC XCCDF result components SHALL validate against the corresponding Schematron rules for FDCC XCCDF reporting available at http://scap.nist.gov/revision/1.2/#schematron.

[23] The FDCC XCCDF results format is described in detail at http://nvd.nist.gov/fdcc/fdcc_reporting.cfm.

4.6 OVAL Results

Each SCAP OVAL result data stream component SHALL use the *<oval-res:oval_results>* element as the document element. Each OVAL result data stream component SHALL validate against version 5.10 of the OVAL Results schema[24] regardless of the version of the OVAL definitions document that was evaluated.

An SCAP OVAL result data stream component SHALL include the results of every OVAL definition used to generate the reported results.

In order to be SCAP conformant, an SCAP content consumer SHALL be able to produce all the types of OVAL Results output described below. The specific result output SHALL be configurable within the SCAP content consumer.

In order to support SCAP instances where OVAL thin content (only the ID of the definition and the results) is preferred, SCAP content consumers SHALL support all valid values for the *<oval-res:directives>* controlling the expected content of the results file.

To support the ability for results to be consumed by the appropriate product(s), data results SHALL be expressed as Single Machine Without System Characteristics, Single Machine With System Characteristics, or Single Machine With Thin Results as follows:

1. Single Machine Without System Characteristics – A single result file that includes all OVAL definitions evaluated and "full" results types as described in the ContentEnumeration element of the OVAL Results schema[25], without system characteristics.

   For this format, the values for the `<oval-res:directives>` element SHALL be:

   ```
   <oval-res:definition_true content="full" reported="true"/>
   <oval-res:definition_false content="full" reported="true"/>
   <oval-res:definition_unknown content="full" reported="true"/>
   <oval-res:definition_error content="full" reported="true"/>
   <oval-res:definition_not_evaluated content="full" reported="true"/>
   <oval-res:definition_not_applicable content="full" reported="true"/>
   ```

2. Single Machine With System Characteristics – A single result file that includes all OVAL definitions evaluated and "full" results types as described in the ContentEnumeration element of the OVAL Results schema and the System Characteristics of the target evaluated.

   For this format, the values for the `<oval-res:directives>` element SHALL be:

   ```
   <oval-res:definition_true content="full" reported="true"/>
   <oval-res:definition_false content="full" reported="true"/>
   <oval-res:definition_unknown content="full" reported="true"/>
   <oval-res:definition_error content="full" reported="true"/>
   <oval-res:definition_not_evaluated content="full" reported="true"/>
   <oval-res:definition_not_applicable content="full" reported="true"/>
   ```

   When creating the OVAL System Characteristics as defined by the `<oval-sc:oval_system_characteristics>` element, the `<oval-sc:collected_objects>` and

[24] The OVAL schemas are described in detail at http://oval.mitre.org/language/about.
[25] The OVAL Results schema is described at http://oval.mitre.org/language/about/structure.html#results.

<oval-sc:system_data> elements SHALL be provided.

3. Single Machine With Thin Results – A single result file that includes all OVAL definitions evaluated and "thin" results types as described in the OVAL Results schema. A value of "thin" means only the minimal amount of information will be provided.

   For this format, the values for the <oval-res:directives> element SHALL be:

```
<oval-res:definition_true content="thin" reported="true"/>
<oval-res:definition_false content="thin" reported="true"/>
<oval-res:definition_unknown content="thin" reported="true"/>
<oval-res:definition_error content="thin" reported="true"/>
<oval-res:definition_not_evaluated content="thin" reported="true"/>
<oval-res:definition_not_applicable content="thin" reported="true"/>
```

When specifying OVAL system characteristics, a reference SHOULD be made to the target asset in the ARF report collection. Specifically, the <oval-sc:oval_system_characteristics>/<oval-sc:system_info>/##any SHOULD be populated with a <ds:asset-identification> element. That element MUST be populated with a single <arf:object-ref> that points to the asset identification element in the ARF report collection pertaining to the OVAL result. See [ARF] for details on populating the <arf:object-ref> element.

4.7 OCIL Results

An SCAP OCIL result data stream component SHALL include the results of every <ocil:questionnaire>, <ocil:question_test_action>, and <ocil:question> element used to generate the reported results.

4.8 Result Data Stream Signing

Digitally signing result data stream content is important to ensuring the integrity and trustworthiness of results. Leveraging [TMSAD] for SCAP can improve the legitimacy of results of SCAP content and create a more secure environment. As such, content consumers MAY digitally sign result content following the guidelines in [TMSAD], along with the following requirements.

One XML digital signature MAY be included in an *<arf:extended-info>* element in the ARF report. The signature MUST be represented as a *<dsig:Signature>* and MUST follow the W3C recommendation [DSIG]. The *<dsig:Signature>* MUST sign the ARF report collection root element.

The *<dsig:Signature>* element MUST follow the recommendations in [TMSAD] and these additional requirements:

1. A *<dsig:SignatureProperties>* MUST be included in the *<dsig:Signature>*. The *<dsig:SignatureProperties>* MUST be populated in accordance with the guidelines in [TMSAD].

2. The first *<dsig:Reference>* in *<dsig:Signature>* MUST be to the root element of the ARF report collection. The element MUST be referenced in the *@URI* attribute using the empty string convention "".

3. Two XPath Filter 2 transforms MUST exist on the first *<dsig:Reference>* in *<dsig:Signature>*. Both MUST specify a filter type of "subtract". The first transform MUST specify the XPath "/arf:asset-report-collection/arf:extended-infos[count(arf:extended-info[dsig:Signature]) = count(*)]". The second transform MUST specify the XPath "/arf:asset-report-collection/arf:extended-infos/arf:extended-info[dsig:Signature]". In both cases, the namespace prefix "arf" MUST map to the ARF namespace specified in this document.

4. The second *<dsig:Reference>* MUST be to the *<dsig:SignatureProperties>* captured in a *<dsig:Object>* with the *<dsig:Signature>*. The *<dsig:SignatureProperties>* MUST be referenced in the *@URI* attribute using "#" + *@Id* of the *<dsig:SignatureProperties>*.

5. Key information SHOULD be provided on the *<dsig:Signature>* in accordance with [TMSAD].

In situations where it is desirable to countersign a result data stream (e.g., when a content consumer automatically signs a result data stream and then a person also wants to sign the results), the following requirements apply.

1. The *<arf:extended-info>* element containing the original signature SHALL be removed from the resulting document.

2. The original signature SHALL be captured as a *<dsig:Object>* on the new *<dsig:Signature>*.

3. The first *<dsig:Reference>* on the new *<dsig:Signature>* SHALL reference the *<dsig:Object>* containing the original signature. The *<dsig:Object>* MUST be referenced in the *@URI* attribute using "#" + *@Id* of the *<dsig:Object>*.

4. The second *<dsig:Reference>* MUST be to the *<dsig:SignatureProperties>* captured in a *<dsig:Object>* with the *<dsig:Signature>*. The *<dsig:SignatureProperties>* MUST be referenced in the *@URI* attribute using "#" + *@Id* of the *<dsig:SignatureProperties>*.

5. A *<dsig:SignatureProperties>* MUST be included in the *<dsig:Signature>*. The *<dsig:SignatureProperties>* MUST be populated in accordance with the guidelines in [TMSAD].

6. Key information SHOULD be provided on the *<dsig:Signature>* in accordance with [TMSAD].

7. The new *<dsig:Signature>* MUST be placed in a new *<arf:extended-info>* element in the ARF report collection.

A signature that has countersigned another signature (also known as an enveloping signature) MAY be countersigned. When doing so, the requirements above SHALL apply to the new signature creation.

When signing a result data stream, the source data stream collection SHOULD be captured in the ARF report being signed.

5. Source Data Stream Content Requirements for Use Cases

This section discusses additional requirements for four SCAP-conformant content use cases: compliance checking, vulnerability scanning, inventory scanning, and OVAL-only scanning. Note that as stated in Table 3 in Section 3.1, each data stream is required to have a `@use-case` attribute in its `<ds:data-stream>` element with a value corresponding to one of the content types defined in this section. The required value for each content type is specified below in the appropriate subsection.

5.1 Compliance Checking

SCAP content can be used to compare system characteristics and settings against an SCAP-conformant checklist in an automated fashion. This can verify that operating systems and applications comply with security checklists and identify any deviations from those checklists.

The SCAP source data stream component that MUST be included for compliance checking is the XCCDF Benchmark, which expresses the checklist. Each rule in the XCCDF Benchmark SHALL reference one of the following:

- An OVAL compliance definition. This definition SHALL be contained in an OVAL component, which holds definitions of compliance checks used by the checklist. An XCCDF Benchmark's rules MAY reference one or more OVAL compliance class definitions in an OVAL component.
- An OCIL questionnaire. This questionnaire SHALL be contained in an OCIL Questionnaire component, which holds questionnaires that collect information that OVAL is not being used to collect, such as posing questions to users or harvesting configuration information from an existing database. An XCCDF Benchmark's rules MAY reference one or more OCIL questionnaires in an OCIL Questionnaire component.
- An OVAL patch definition. This definition SHALL be contained in an OVAL component, which holds definitions for patch compliance checks. These checks may be needed if an organization includes patch verification in its compliance activities. An XCCDF Benchmark MAY reference an OVAL patch definition through a patches up-to-date rule in a manner consistent with Section 3.3.6.4.

Each XCCDF Benchmark SHALL have at least one rule that references either an OVAL compliance class definition in an OVAL component or an OCIL questionnaire in an OCIL Questionnaire component.

All OVAL components and OCIL Questionnaire components referenced by the XCCDF Benchmark SHALL be included in the SCAP source data stream.

If the XCCDF Benchmark component references any CPE names, then the SCAP source data stream MUST include the CPE Dictionary component, which specifies the products or platforms of interest, and MUST include one or more OVAL inventory class definitions in an OVAL component that contain the technical procedures for determining whether or not a specific target asset has a product or platform of interest.

The `@use-case` attribute in the `<ds:data-stream>` element MUST be set to "CONFIGURATION".

5.2 Vulnerability Scanning

SCAP content can be used to scan operating systems and applications to look for known software flaws that introduce security exposures. The content enables consistent detection and reporting of these flaws.

The SCAP source data stream component that MUST be included for vulnerability scanning is the XCCDF Benchmark, which expresses the checklist of the flaws to be checked for. Each rule in the XCCDF Benchmark SHALL reference one of the following:

- An OVAL vulnerability definition. This definition SHALL be contained in an OVAL component, which holds definitions of vulnerability checks used by the checklist. An XCCDF Benchmark's rules MAY reference one or more OVAL vulnerability class definitions in an OVAL component.
- An OCIL questionnaire. This questionnaire SHALL be contained in an OCIL Questionnaire component, which holds questionnaires that collect information that OVAL is not being used to collect, such as giving a system administrator step-by-step directions for manually examining a system for a vulnerability that cannot be detected with OVAL, and then collecting information on the results of that manual examination. An XCCDF Benchmark's rules MAY reference one or more OCIL questionnaires in an OCIL Questionnaire component.
- An OVAL patch definition. This definition SHALL be contained in an OVAL component, which holds definitions for patch compliance checks. These checks may be needed if an organization includes patch verification in its vulnerability scanning activities. An XCCDF Benchmark MAY reference an OVAL patch definition through a patches up-to-date rule in a manner consistent with Section 3.3.6.4.

Each XCCDF Benchmark SHALL have at least one rule that references either an OVAL vulnerability class definition in an OVAL component or an OCIL questionnaire in an OCIL Questionnaire component.

All OVAL components and OCIL Questionnaire components referenced by the XCCDF Benchmark SHALL be included in the SCAP source data stream.

If the XCCDF Benchmark component references any CPE names, then the SCAP source data stream MUST include the CPE Dictionary component, which specifies the products or platforms of interest, and MUST include one or more OVAL inventory class definitions in an OVAL component that contain the technical procedures for determining whether or not a specific target asset has a product or platform of interest.

The `@use-case` attribute in the `<ds:data-stream>` element MUST be set to "VULNERABILITY".

5.3 Inventory Scanning

SCAP content can be used to collect information on the software installed on systems. One example of how this could be used is to verify that a group of systems all have required security software programs installed. This could help verify compliance with technical security control requirements. Another example is to collect software inventory data on devices that are not directly connected to the enterprise network, such as smart phones.

Inventory scanning can also be applied to collect information on the presence of software artifacts on systems, such as malware or characteristics of malware that indicate its presence. SCAP content authored for this purpose can be used to detect classes or categories of malware based on system state that may be common across multiple malware instances. For example, it is a common practice to reuse malware code, making modifications to address available detection methods, change propagation characteristics, etc. It is also possible to author content that detects a specific instantiation of malware. For example, hashing of files can be used to identify a malicious executable or library.

The SCAP source data stream components that MUST be included for inventory scanning is the XCCDF Benchmark, which references the inventory checks and captures the results. Each rule in the XCCDF Benchmark SHALL reference one of the following:

- An OVAL inventory definition. This definition SHALL be contained in an OVAL component, which holds definitions of technical procedures for determining whether or not a specific target asset has software (product, platform, malware, etc.) of interest. An XCCDF Benchmark's rules MAY reference one or more OVAL inventory class definitions in an OVAL component.
- An OCIL questionnaire. This questionnaire SHALL be contained in an OCIL Questionnaire component, which holds questionnaires that collect information that OVAL is not being used to collect, such as posing questions to users or harvesting inventory information from an existing database. An XCCDF Benchmark's rules MAY reference one or more OCIL questionnaires in an OCIL Questionnaire component.

The `@use-case` attribute in the `<ds:data-stream>` element MUST be set to "INVENTORY".

5.4 OVAL-Only Scanning

OVAL content can be used on its own, without XCCDF Benchmarks or other components, to perform scanning. The only SCAP source data stream component that MUST be included is an OVAL component with the desired definitions (e.g., compliance class for configuration setting checks, inventory class for asset checks, patch class for patch presence checks, vulnerability class for software flaw vulnerability presence checks). The mapping SHALL correspond to the mappings defined in Section 3.3.6.2.

The `@use-case` attribute in the `<ds:data-stream>` element MUST be set to "OVAL_ONLY".

Content consumers SHOULD support a mechanism to process standalone OVAL content that is not part of a source data stream. Content processed in this way would not take advantage of the capabilities provided by the source data stream format.

Appendix A—Acronyms and Abbreviations

Selected acronyms and abbreviations used in the guide are defined below.

API	Application Programming Interface
ARF	Asset Reporting Format
CCE	Common Configuration Enumeration
CCSS	Common Configuration Scoring System
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DHS	Department of Homeland Security
DoD	Department of Defense
FISMA	Federal Information Security Management Act
IR	Interagency Report
IT	Information Technology
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OCIL	Open Checklist Interactive Language
OMB	Office of Management and Budget
OS	Operating System
OVAL	Open Vulnerability and Assessment Language
PCI	Payment Card Industry
RFC	Request for Comments
SCAP	Security Content Automation Protocol
SP	Service Pack
SP	Special Publication
TMSAD	Trust Model for Security Automation Data
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XCCDF	Extensible Configuration Checklist Description Format
XML	Extensible Markup Language

Appendix B—Normative References

The following normative references are pointers to the specifications, schema, dictionaries, and other information that are required to implement the SCAP 1.2 components:

[AI] Asset Identification spec., description http://scap.nist.gov/revision/1.2/#ai

[ARF] ARF specification and description http://scap.nist.gov/revision/1.2/#arf

[CCE] CCE specification and description http://scap.nist.gov/revision/1.2/#cce

[CCSS] CCSS specification and description http://scap.nist.gov/revision/1.2/#ccss

[CPE] CPE specifications and description http://scap.nist.gov/revision/1.2/#cpe

[CPE-D] CPE Dictionary specification http://scap.nist.gov/specifications/cpe/#dictionary

[CPE-L] CPE Applicability Language spec. http://scap.nist.gov/specifications/cpe/#language

[CPE-M] CPE Matching specification http://scap.nist.gov/specifications/cpe/#matching

[CPE-N] CPE Naming specification http://scap.nist.gov/specifications/cpe/#naming

[CVE] CVE specification and description http://scap.nist.gov/revision/1.2/#cve

[CVSS] CVSS specification and description http://scap.nist.gov/revision/1.2/#cvss

[DSIG] DSIG specification and description http://www.w3.org/TR/xmldsig-core/

[ERRATA] SCAP 1.2 (SP 800-126) errata http://scap.nist.gov/revision/1.2/#errata

[OCIL] OCIL specification and description http://scap.nist.gov/revision/1.2/#ocil

[OVAL] OVAL specification and description http://scap.nist.gov/revision/1.2/#oval

[TMSAD] Trust Model spec. and description http://scap.nist.gov/revision/1.2/#tmsad

[XCCDF] XCCDF specification and description http://scap.nist.gov/revision/1.2/#xccdf

[XLINK] XLink specification http://www.w3.org/TR/xlink/

[XMLCAT] XML Catalog specification
http://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html

[XMLS] W3C XML Schema http://www.w3.org/XML/Schema.html

Appendix C—Change Log

Release 0 – 12 July 2011

- Complete draft specification released for public comment.